InterWorking Labs®

Are you Allocating your Test Resources Correctly?

The Importance of Functional Testing

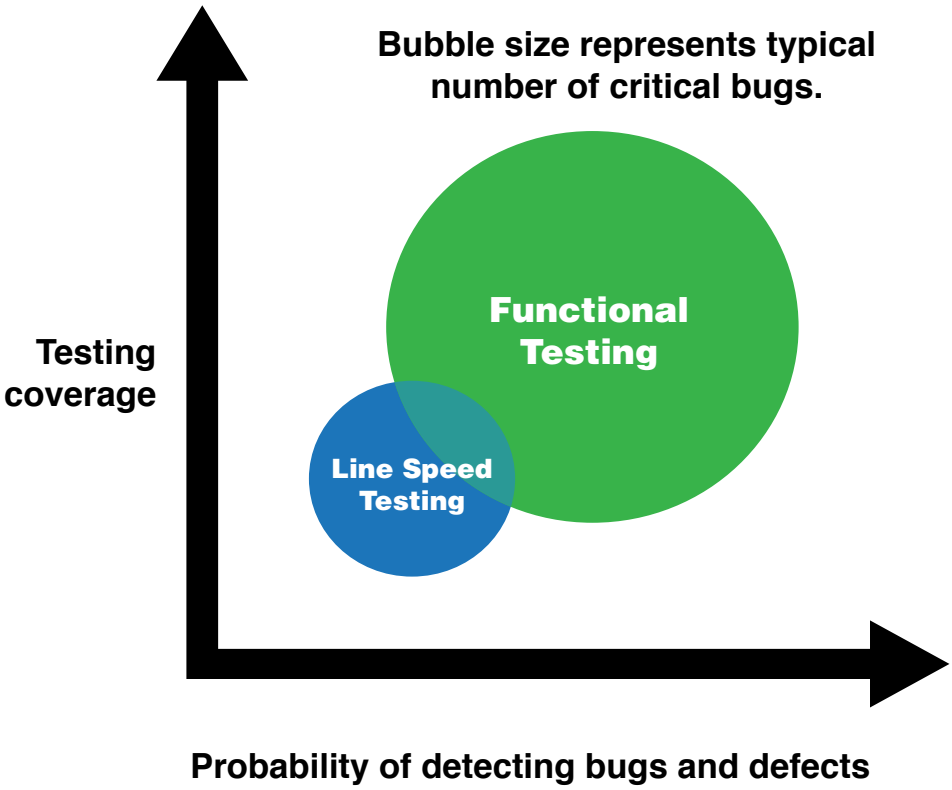
Both network applications and network equipment undergo rigorous testing before shipment to customers, and deployment on the Internet. Assuring reliable and deterministic behavior under the broadest possible range of conditions is the goal of all network equipment testing. In a vast and unpredictable environment like the Internet, your customers expect a network device to perform and remain stable regardless of network conditions. Whether a hacker generates millions of queries to find vulnerabilities, creates a DoS attack, or a software developer simply forgets a critical semicolon in the code, you need to guard against device failure, through design, development, testing, and release.

Many software developers and quality assurance engineers associate testing with expensive packet generators. These test systems send millions of identical packets at your device so the quality assurance engineer can observe the effects of an overloaded system. This "line speed testing" may help characterize the throughput of a network device, but that's the extent of the information. The quality assurance engineer learns, for example, that the 20 Gbps interface on the device can only accommodate 19 Gbps! He completes his report, and sends the system back to the development engineers for more fine tuning and engineering. While line speed testing has a role – the product must operate at its rated speed – it is a minor role. Why? Because the testing imperative is reliability.

What's Inside...

InterWorking Labs®

Kings Village Center #66190
Scotts Valley, CA 95067
iwl.com
+1.831.460.7010
info@iwl.com



The Paradox

The difference between line speed testing and functional testing has brought about a paradox in the world of system test. While functional testing tends to deliver the greater returns (e.g. finding more critical bugs), line speed testing often gets prioritized.

For most quality assurance engineers, line speed testing is easier to understand. In our earlier example, the benchmark was 20 Gbps sustained throughput. Line speed testing demonstrates that the system currently delivers 19 Gbps. Do the math! It is one Gbps short of the requirement. That's easy to understand.

In functional (conformance and compliance) testing, a quality assurance engineer may get a more complex test result[1] that requires the engineer to analyze the error, possibly consult some computer science texts on its properties, and try to understand how or why it occurred. That's a lot of work!

Compare that analysis to understanding that 19 Gbps is less than 20 Gbps, and the appeal of line speed testing becomes clear.

Line Speed Testing	Functional Testing
19 Gbps < 20 Gbps. Our product is one Gbps too slow!	The response OID 1.3.6.1.2.1.4.24.4.1.1.0.0.0.0.0.0.0.0.0.0.199.184.129.1 is lexicographically less than or equal to the request OID 1.3.6.1.2.1.4.24.4.1.1.0.0.0.0.4294967295.0.0.0.0.0.199.184.129.1

Which column is easier to read?

The example illustrates how quality assurance engineers may favor line speed testing and give it more attention than warranted.

Line speed testing also frequently requires expensive specialized test hardware which means it often gets the lion's share of the test equipment budget. However, it is functional testing that will most often uncover the most insidious bugs and architectural defects.

Realize that functional tests exercise more of your code. Exception cases and unusual conditions aren't caught when just pumping the same packet through the system repeatedly. It takes functional tests to exercise the boundary conditions, and it is this boundary testing that truly hardens products before they ship.

The result is that an investment in functional test tools will typically pay for itself much faster than the return on line speed testing. The payback on functional testing has been documented through third-party research. Fixing a single defect once a product has shipped can be as high as 10-times the cost of fixing the problem before product release.[2]

Why Line Speed Testing Isn't Enough

By focusing on line speed testing, companies often leave themselves more exposed than they realize.

Think of it this way - what is the value of throwing more traffic at a device than you know it can handle? You already know the device will not be able to process all the packets; all you can accomplish is to verify that the device fails gracefully. While important, this is a small component of the testing required.

Functional testing, on the other hand, tests the broad range of use cases that can cause a device to fail or act unpredictably. These types of bugs are much harder to trace back to a root cause when first reported by your customers. Imagine your top customer experiencing an unexplained and infrequent crash that you cannot reproduce in your lab. Your customer loses faith in your product and your reputation suffers while you spend weeks or months trying to identify a root cause. Often the cause of the fault can be as simple as a single errant malformed packet that triggers a software defect (something that would be uncovered with proper functional testing). If the customer had reported a crash due to overwhelming the device with excessive traffic, the root cause would be much simpler to find and fix.

The Hidden Costs of Misallocated Test Resources

Preemptive testing of boundary conditions can help your engineers find and fix errant problems before they are discovered in the field. The cost of a test suite to exercise a broad range of abnormal conditions is generally far less than a team of engineers trying to track down a problem reported by a customer.[3] Add to that the cost of losing your customer's confidence or business, and it quickly becomes apparent that the investment in functional testing is worth every penny.

In short, functional testing uncovers the more subtle bugs. These are the bugs that are hardest to find once your product is deployed. All devices require functional testing, whereas only specialized packet forwarding devices (such as network switches and routers) really benefit from intensive line speed testing.

Comprehensive functional testing will save—and can even help make—you money by:

- Reducing up-front costs (functional tests are more economical than specialized line speed testing hardware)
- Minimizing expense on the back-end by eliminating the fire fighting that would have occurred had the product shipped with defects
- Maintaining and enhancing your reputation with your customers, preserving the relationship for future business opportunities.

Build vs. Buy

No business can provide 100% perfect test coverage. You need to spend your dollars wisely and get the greatest bang-for-the-buck. A frequent mistake is to assume that it is cheaper to hire a low-wage engineer to write tests than it is to buy a test suite. Remember, coverage is the key.

Third-party test suites have been expanded and matured over time, whereas internally developed tests invariably miss large segments of testing due to limited time, inexperience and the myopic views of internal development. When the same development team is deciding what to test, big areas are often overlooked simply due to "group-think". Only massive programs (like the space shuttle program) can afford to separately staff a completely independent and comprehensive test program.[4] And even these programs are smart enough to buy whatever tests they can in order to maximize coverage and to concentrate on building only what cannot be purchased.

Summary

In short, spend wisely but don't overlook the costs of releasing a buggy product. Finding and fixing software defects constitutes the largest identifiable expense for the software industry.[5]

Your reputation is on the line every time your product goes into the field. Reputations take years to build, and only moments to destroy. Strategic product testing is the key to building and maintaining the reputation of both your products and your company.

Notes

[1] Example of Test Results from Functional Testing

> [FAILED] Remarks: get-next operation failed or had errors Lexicographic error detected in response to get-next request sent.
>
> The response OID 1.3.6.1.2.1.4.24.4.1.1.0.0.0.0.0.0.0.0.0.199.184.129.1 is lexicographically less than or equal to the request OID 1.3.6.1.2.1.4.24.4.1.1.0.0.0.4294967295.0.0.0.0.0.199.184.129.1
>
> The consequences of your agent behaving in this way is that the data returned is indeterminate and will likely confuse management applications. You could get duplicates. You are not operating efficiently. You may miss rows in the retrieval. The management application could become so confused that it terminates before retrieving all the objects. The rule is that the respose OID must be lexicographically greater than the request OID.

[2] NIST, "The Economic impacts of Inadquate Infrastructure for Software Testing", 2002, p. 1-13, Table 1-5.
http://www.nist.gov/director/planning/upload/report02-3.pdf

[3] IBID.

[4] Fishman, Charles. "They Write The Right Stuff"
www.fastcompany.com/28121/they-write-right-stuff

[5] Jones, Capers. Software Quality: Analysis and Guidelines for Success. International Thomson Computer Press. Copyright 1997. Page 173.

